ORACLE
Hospitality

Event-driven API strategies

from WebHooks to GraphQL Subscriptions

Luis Weir
Code Monsters
11th December, 2019

Hospitality

About Me

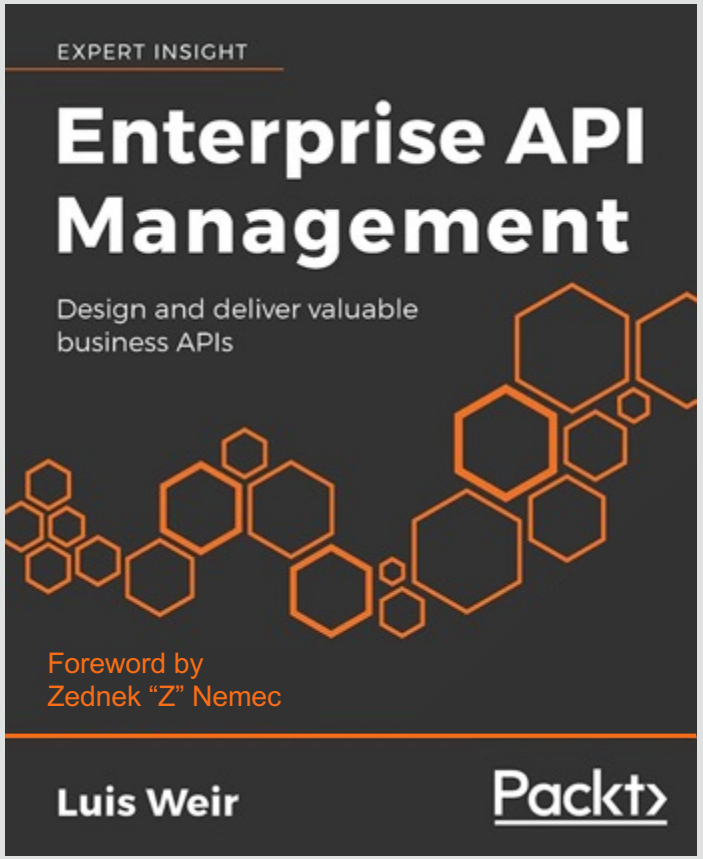
Luis Weir
luis.weir@oracle.com
Director of Technology & Developer

@luisw19

luisw19

www.soa4u.co.uk

BOOKS



tinyurl.com/eapim18 July 2019

tinyurl.com/apim15
Released in Set. 2015

tinyurl.com/soagov13
Released in Set. 2013

apiplatform.cloud/
Released Q2 2018

ARTICLES

- The API lifecycle
- What Is API-Led — An Architectural Approach
- A brief look at the evolution of interface protocols leading to modern APIs
- The 7 Deadly Sins of API Design
- Setting the vision, strategy and direction — the CTO's role
- How can you design, deploy and manage your APIs?
- The Spotify's Engineering Culture. My interpretation and summary
- A comparison of API Gateways communication styles
- Is BPM Dead, Long Live Microservices?
- Five Minutes with Luis Weir
- 2nd vs 3rd Generation API Platforms - A Comprehensive Comparison
- Podcast: Are Microservices and APIs Becoming SOA 2.0?
- 3rd-Generation API Management: From Proxies to Micro-Gateways

This is an opinionated presentation expressing **my own** views.

Hospitality

Agenda

1. **Sync vs Async APIs**
2. Asynchronous on the web?
   Trendy Async API Styles
3. Demo
4. Conclusions & Q & A

Hospitality

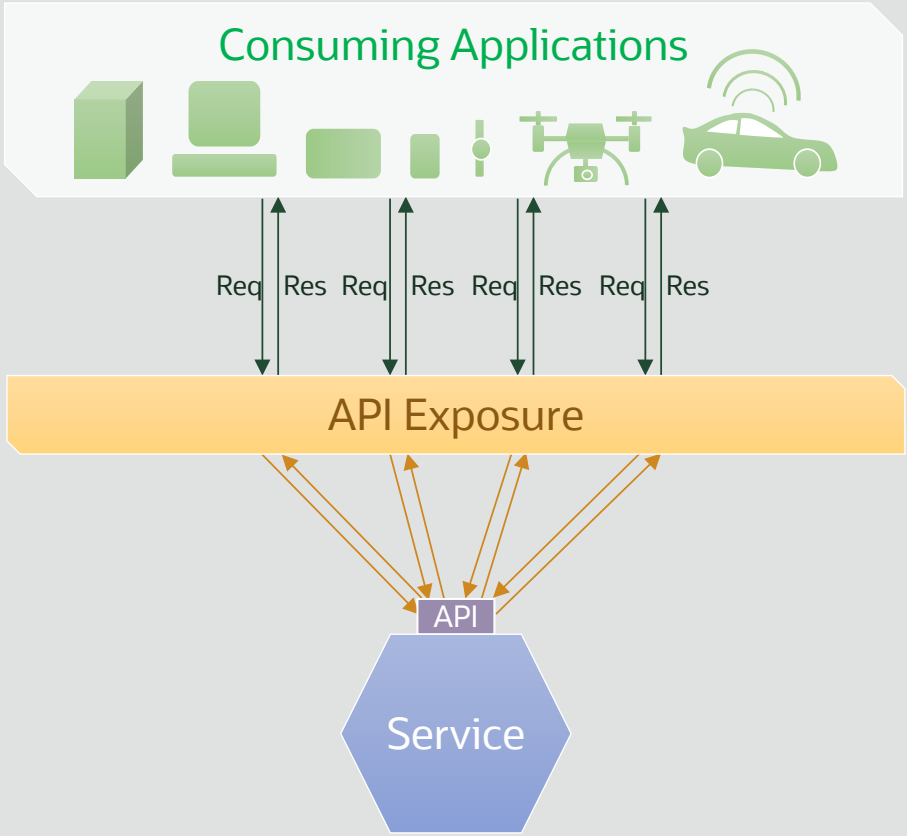
Synchronous communication

Definition of ***Synchronous***

"existing or occurring at the same time"

Synchronous APIs

- **Single http thread** for processing a **request** and a **response**
- **Synchronous** can be very **good** for many **use cases** (instant response needed) but **not so good** when response occurs in a **different time** (event-driven)
- Regular **pulling** as common practice to obtain **updates**



Asynchronous communication

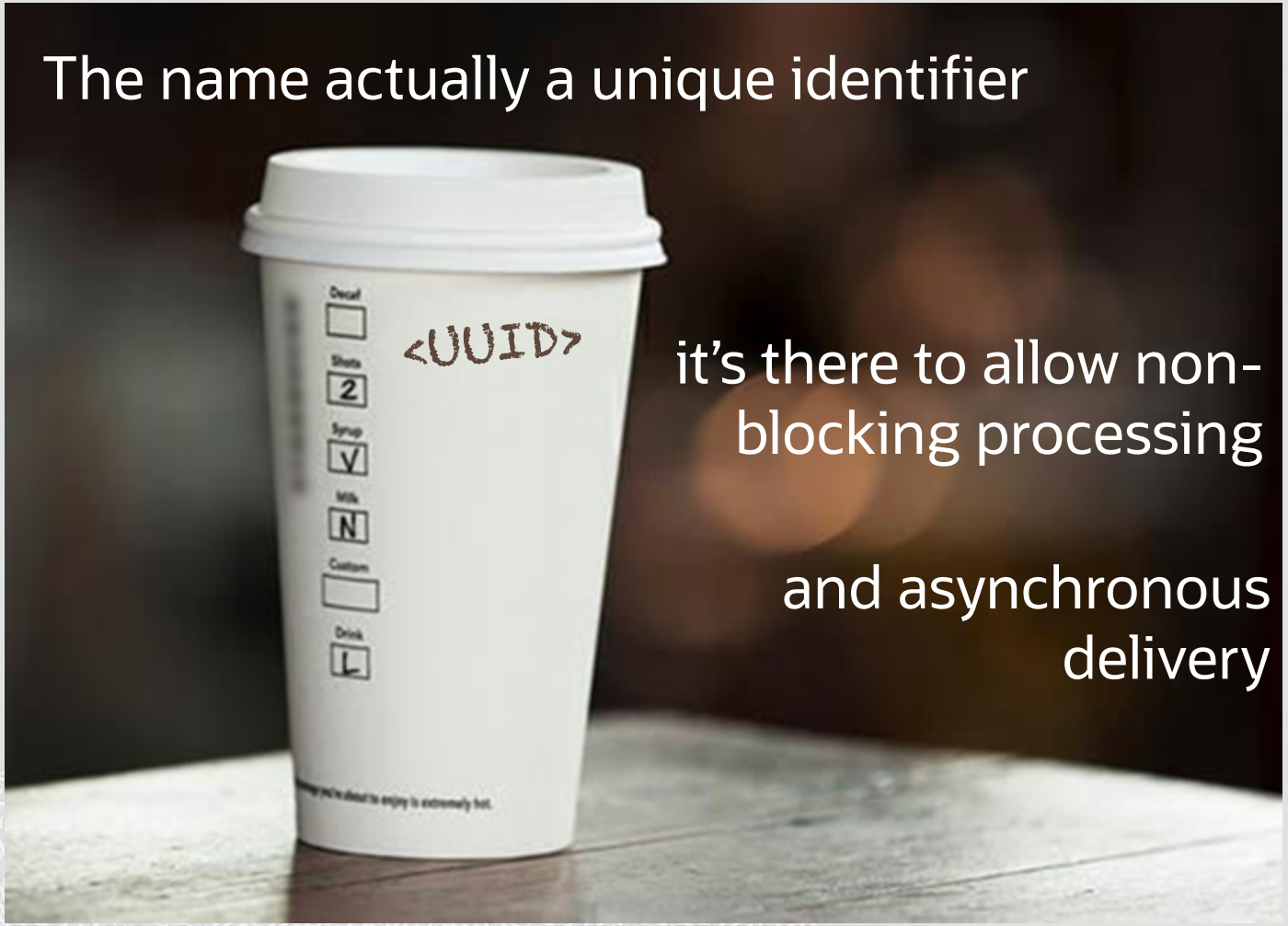
Definition of ***Asynchronous***

"not *existing or occurring at the same time*"

Asynchronous communication

Have you ever thought about the **name in the cup** from a programmers point of view?

Asynchronous communication



Have you ever thought about the **name in the cup** from a programmers point of view?

Think about it....

1) an order is placed under one's name. Then you get out of the queue.
[a command with an UUID]

Think about it....

1) an order is in placed in one's name, then get out of the queue (a command with an UUID)

2) The barrister makes the coffee whilst other orders are placed [non-blocking command executed]

Think about it....

1) an order is in placed in one's name, then get out of the queue (a command with an UUID)

2) the order is processed whilst other orders are placed (non-blocking command exec)

3) The order is delivered by calling one's name again
[a push event is made to fulfilled request]

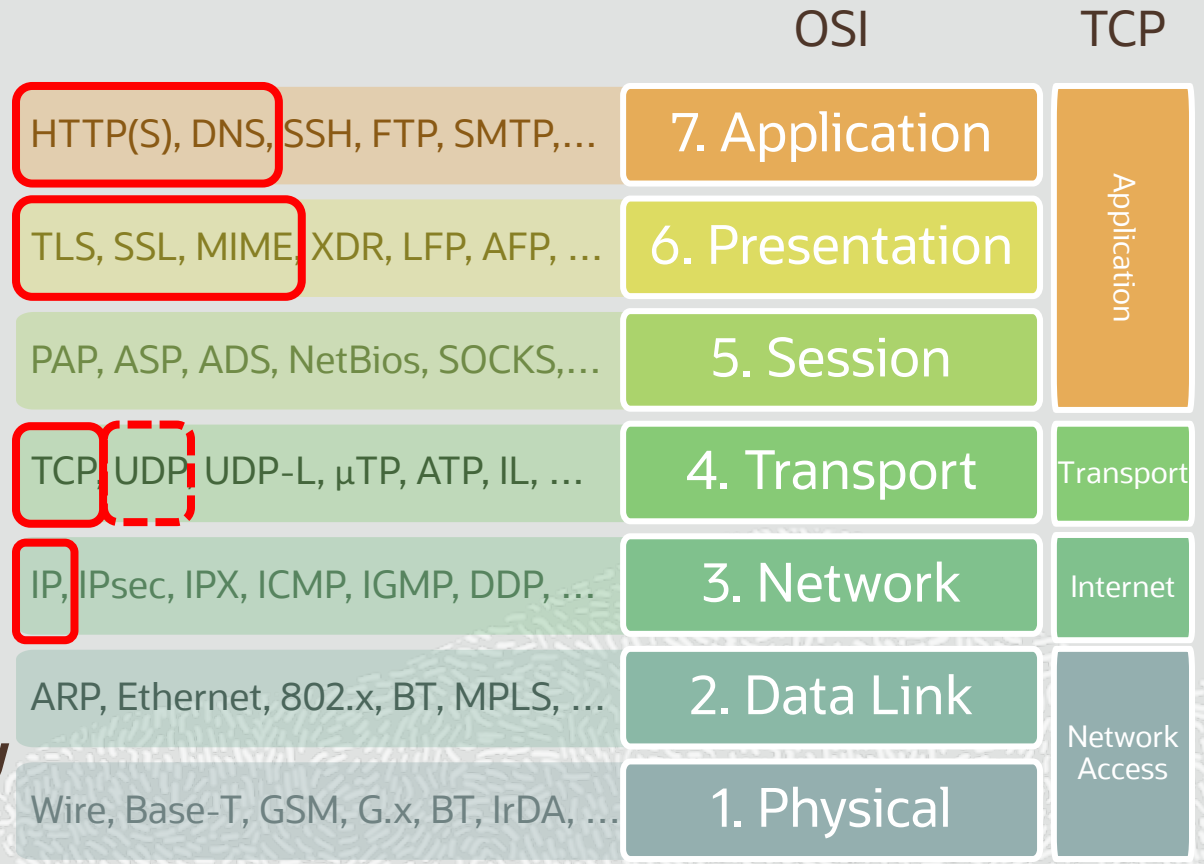
Agenda

1. Sync vs Async APIs
2. **Asynchronous on the web?**
   Trendy Async API Styles
3. Demo
4. Conclusions & Q & A

Hospitality

When talking about Web APIs there are some key considerations to be wary of

- **HTTP(s)** as the **main protocol** supported in **API Gateways** and other **layer 7 appliances** e.g. (e.g. load balancers, web app firewalls, CDNs)
- Majority of corporate **firewalls** (cloud and on-prem) configured to **block non-http(s)** traffic
- **JSON** currently most popular data format in modern APIs -event though for async comms **Avro**, **Protobuf** to name a few picking up fast.
- **HTTP/2** adoption rapidly increasing but still **early days** in the context of API related tech.

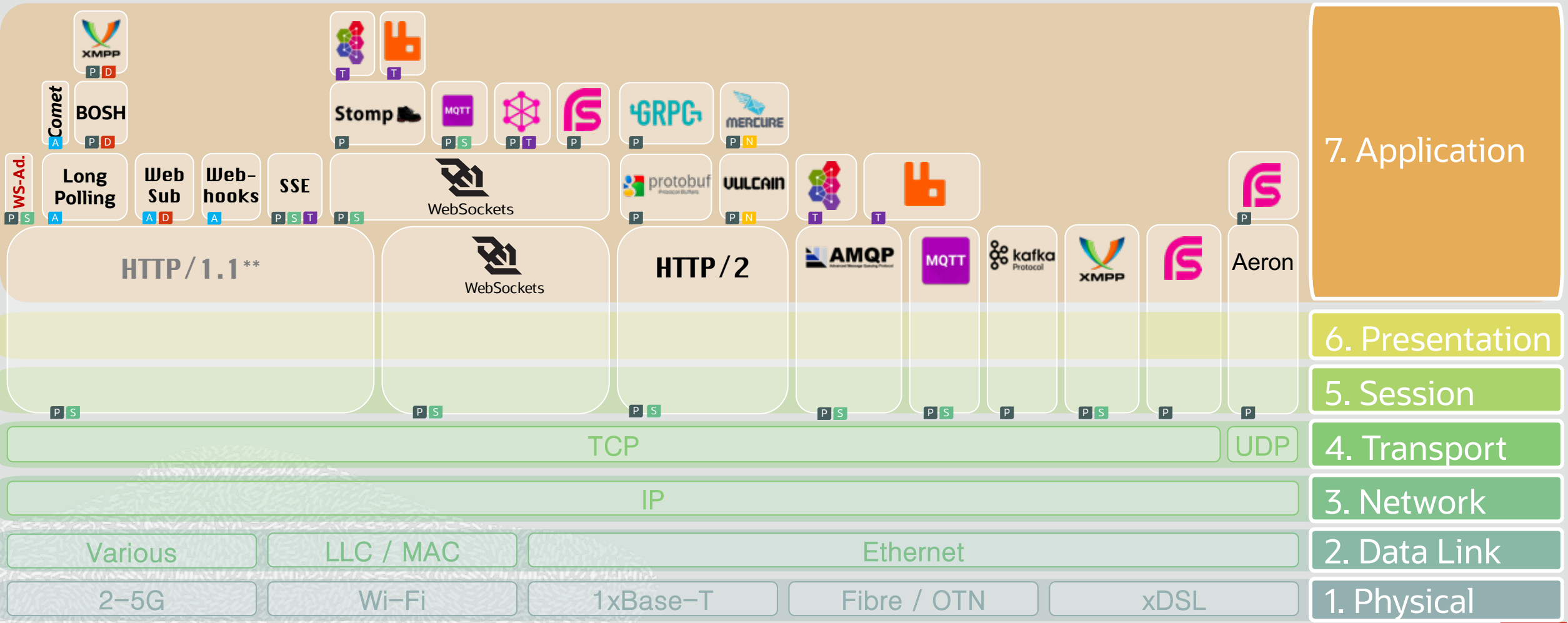


Async technology landscape

Active MQ
Rabbit MQ
GraphQL
rSocket

P Protocol / Specification
S Industry Standard
D Draft or Recommendation
T Tool

A Arch. Pattern / Technique
N New / recent



** no natively asynchronous

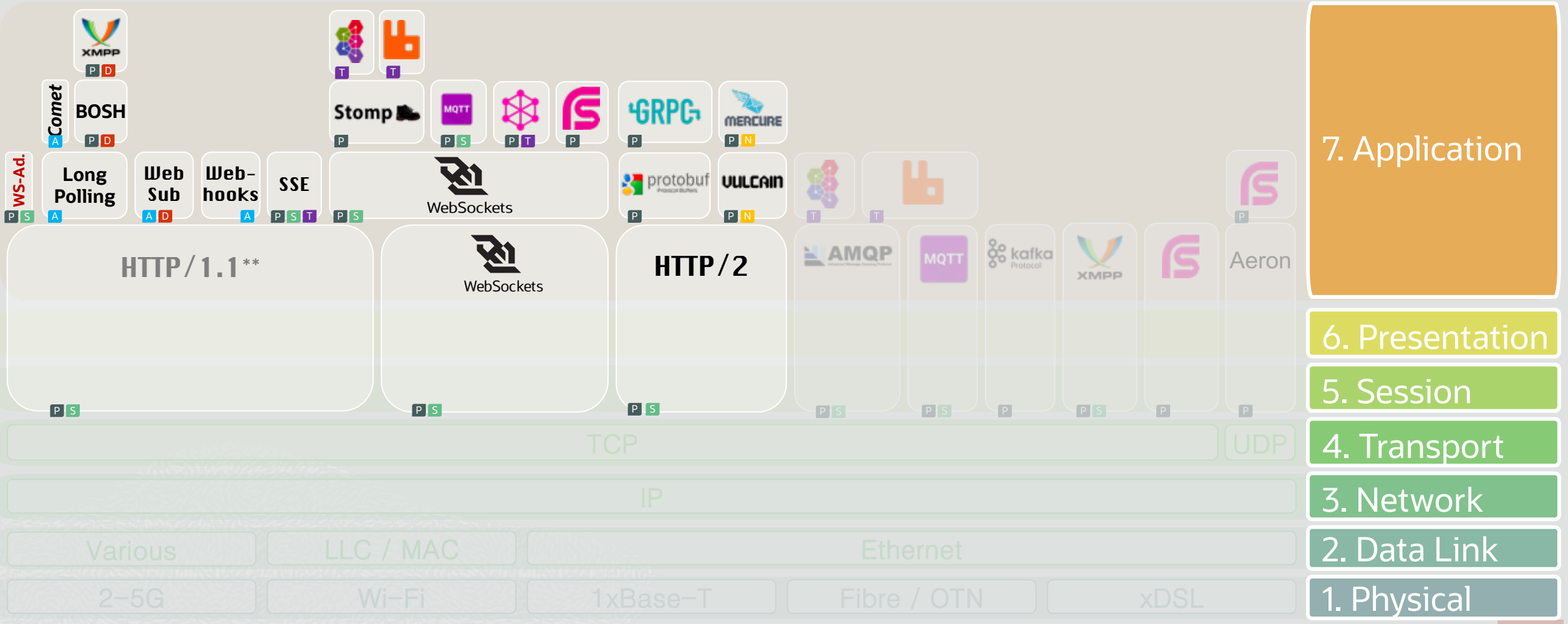
@luisw19

Async technology landscape

Active MQ
Rabbit MQ
GraphQL
rSocket

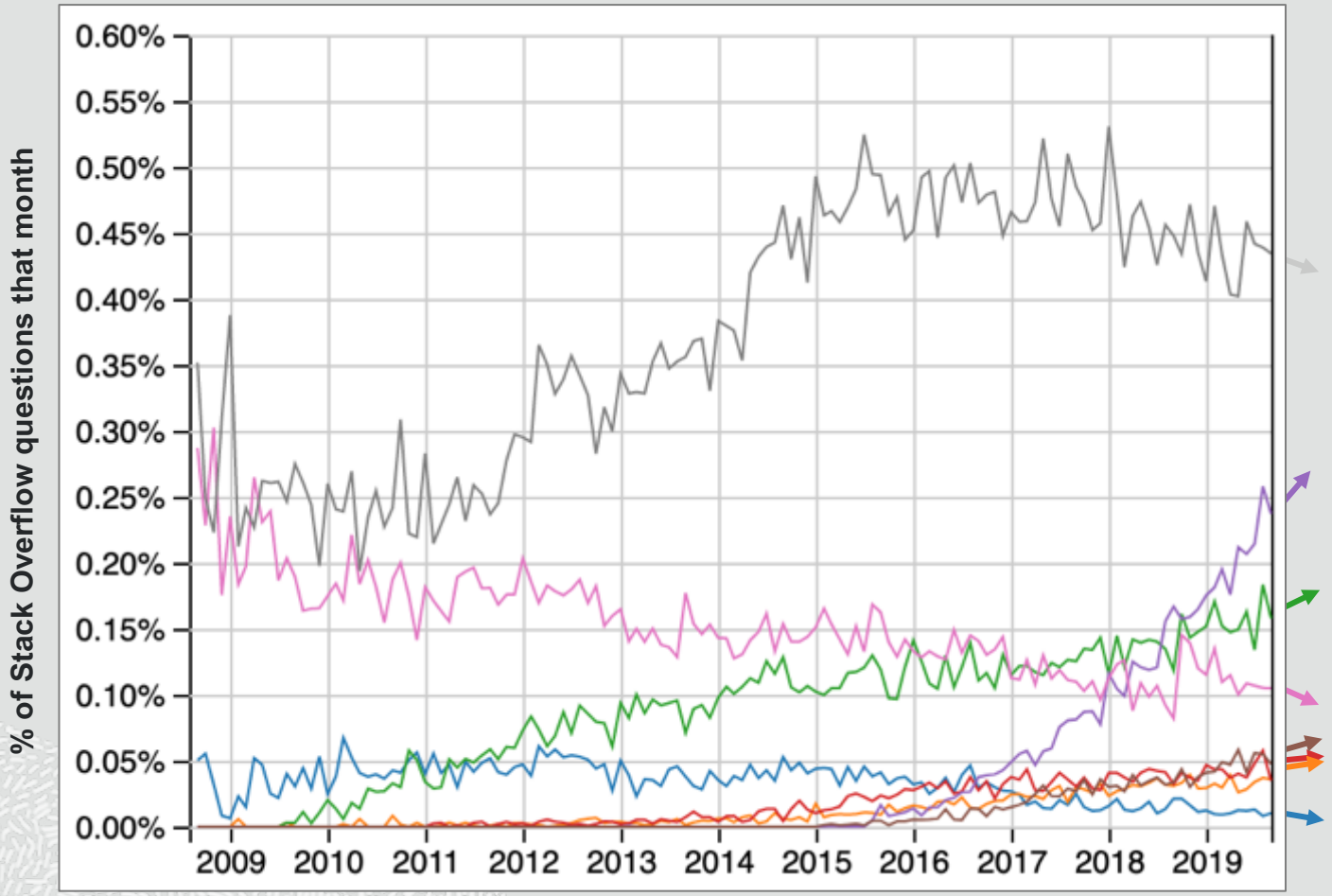
P Protocol / Specification
S Industry Standard
D Draft or Recommendation
T Tool

A Arch. Pattern / Technique
N New / recent



** no natively asynchronous

Async Landscape Trends on Stack Overflow



https://insights.stackoverflow.com/trends?tags=xmpp%2Cwebhooks%2Cwebsocket%2Cmqtt%2Cgraphql%2Cgrpc%2Csoap%2Crest

Trendy Asynchronous API Styles

Long Polling

Webhooks

WebSockets

GraphQL Subscriptions

gRPC

Terminology

Consuming Application

A consuming application of any type unless otherwise explicitly indicated (e.g. browser based app, server side app, etc).

API Exposure

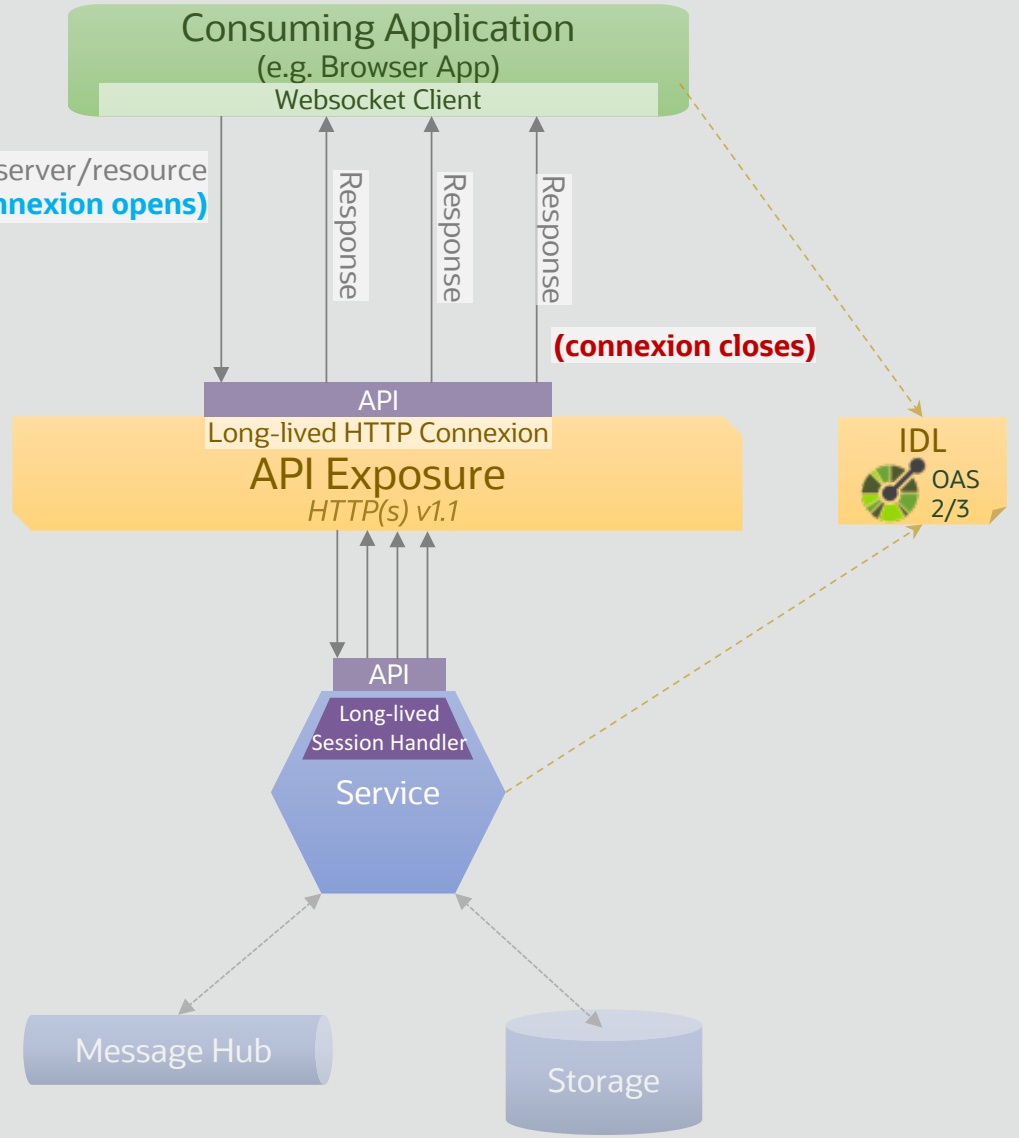
Any layer from which a **service interface** can be **accessed** (e.g. API gateway, ingress, HTTP proxy, load balancer, web firewall, etc)

Service

A (business or technical) **capability encapsulated** in a **service** and that is **accessible** via a **programming interface** (API).

Long Polling APIs

- Consuming application makes **standard HTTP 1.1 request call** however the server **holds** the **response (connexion remains open)**
- As **new event**s occur (e.g. changes in resource) **responses are sent back.**
- Server can response with header "**Transfer-Encoding: chunked**" to avoid creating a new connexions
- **Long-polling/streaming endpoints** can be designed/defined with **OAS 2/3** by defining the relevant **response headers** (e.g. Transfer-Encoding)



Long Pooling Samples

- **Twitter Streaming API:** https://developer.twitter.com/en/docs/tweets/filter-realtime/overview

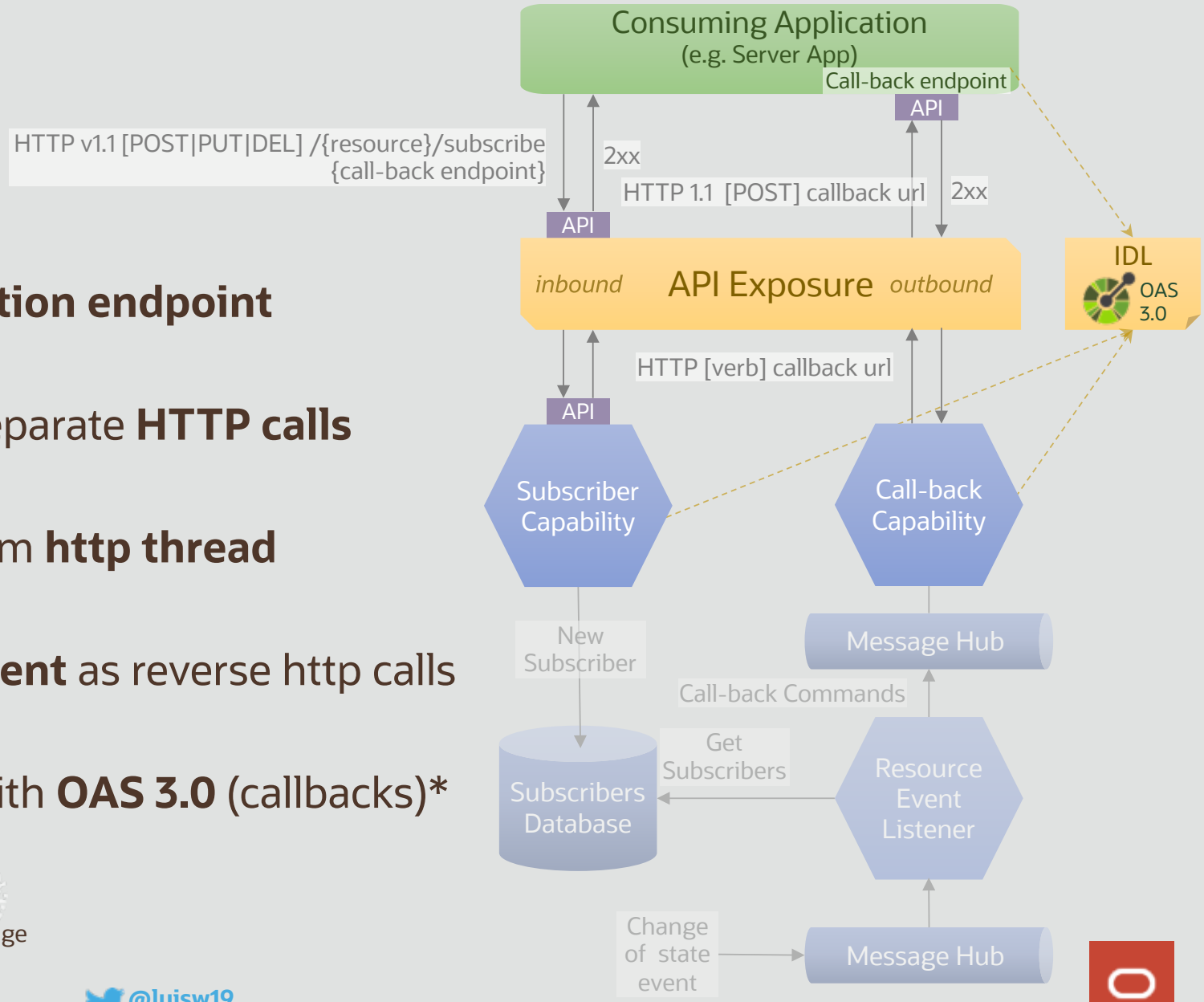
  "Get only the Tweets you need by using advanced filtering tools with the realtime streaming API. BC"

- **SalesForce Streaming API:** https://developer.salesforce.com/docs/atlas.en-us.api_streaming.meta/api_streaming/intro_stream.htm

  "Streaming API enables streaming of events using push technology and provides a subscription mechanism for receiving events in near real time. The Streaming API subscription mechanism supports multiple types of events, including PushTopic events, generic events, platform events, and Change Data Capture events"

Webhook APIs

(akas REST-Hooks)



- **Callback URL** registered via **subscription endpoint**
- **Message request** and **response** in separate **HTTP calls**
- Process **execution** fully **detached** from **http thread**
- Relevant **events** are **pushed** to the **client** as reverse http calls
- **Interface** can be designed/defined with **OAS 3.0** (callbacks)*

* **AsyncAPISpec** working on samples
* **supermodel.io** can be used as as domain (ubiquitous) modelling language

Webhook API Samples

- **GitHub:** https://developer.github.com/webhooks/

  "Webhooks allow you to build or set up integrations, such as GitHub Apps or OAuth Apps, which subscribe to certain events on GitHub.com. When one of those events is triggered, we'll send a HTTP POST payload to the webhook's configured URL. Webhooks can be used to update an external issue tracker, trigger CI builds, update a backup mirror, or even deploy to your production server. You're only limited by your imagination."

- **Paypal:** https://developer.paypal.com/docs/api/webhooks/v1/

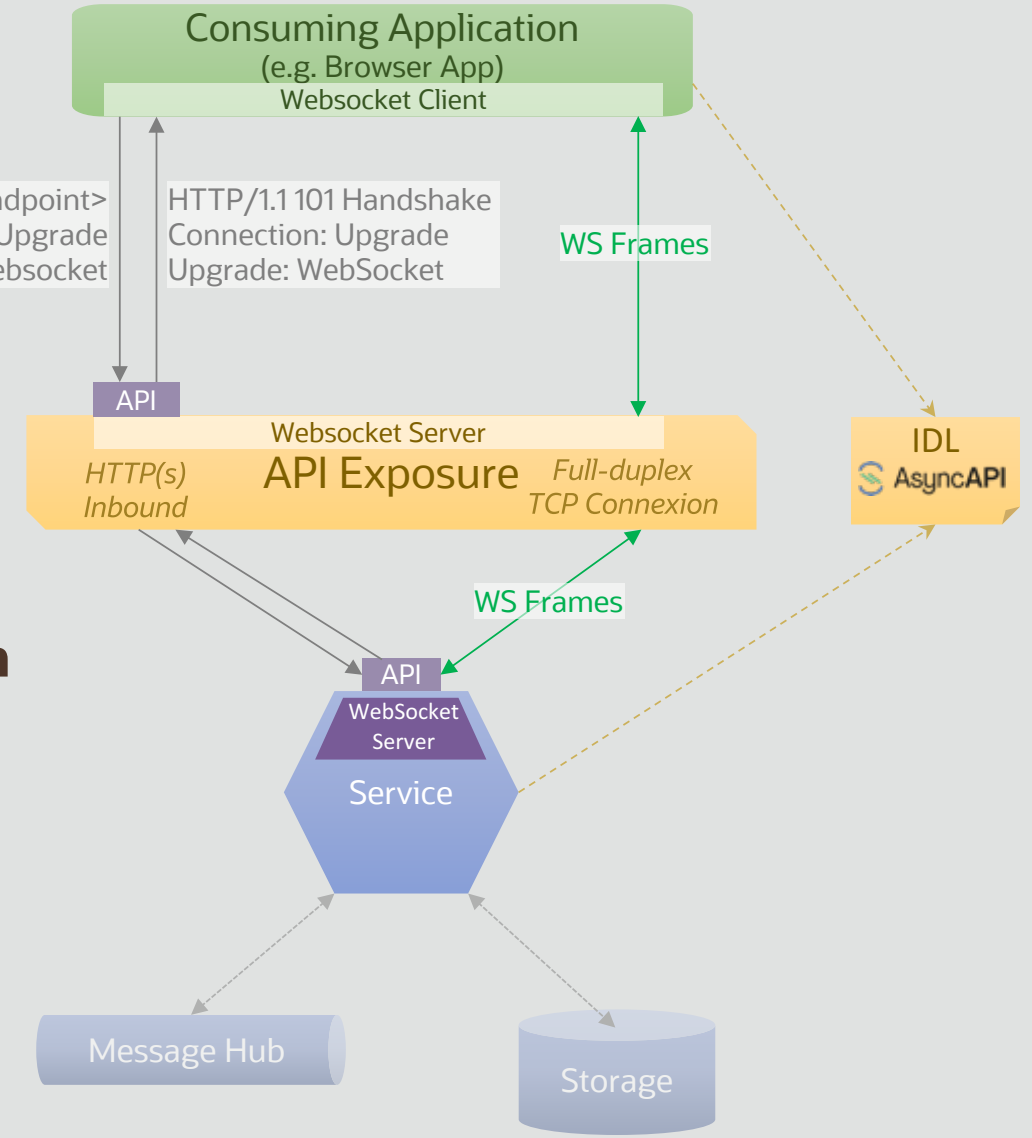
  "The PayPal REST APIs use webhooks for event notification. Webhooks are HTTP callbacks that receive notification messages for events. After you configure a webhook listener for your app, you can create a webhook, which subscribes the webhook listener for your app to events. The notifications namespace contains resource collections for webhooks."

- **W3C:** https://w3c.github.io/w3c-api/webhooks

  "Webhooks allow you to subscribe to certain events happening at W3C. When one of these events is triggered, we'll send a HTTP POST payload to the webhook's configured URL."

WebSocket APIs

- **Client** initiates **WebSocket handshake** to establish connection (HTTP call with upgrade headers)
- Once connexion is upgraded a **full-duplex** communication is **established** via **single TCP connection**
- **WebSocket client** required (majority of **browsers** already **support it**)
- **Interface** can be designed/defined with **AsyncAPISpec***



* **supermodel.io** can be used as as domain (ubiquitous) modelling language

WebSocket API Samples

- **Slack:** https://api.slack.com/rtm

  "The Real Time Messaging API is a WebSocket-based API that allows you to receive events from Slack in real time and send messages as users. It's sometimes referred to as simply the "RTM API""

- **Blockchain.com:** https://www.blockchain.com/api/api_websocket

  "Our WebSocket API allows developers to receive Real-Time notifications about new transactions and blocks"

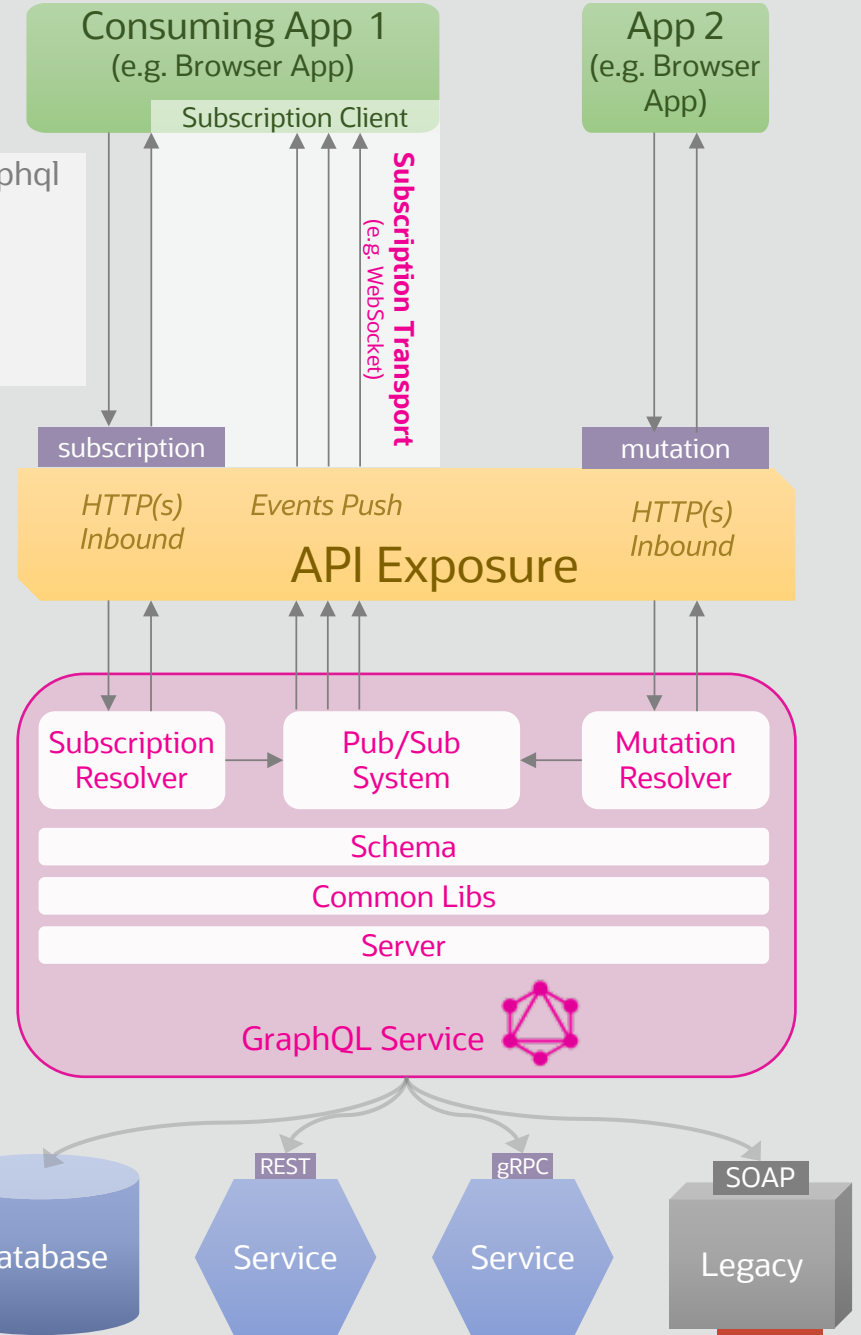
- **FitBit:** https://dev.fitbit.com/build/guides/communications/messaging/

  "The Messaging API allows developers to easily send simple messages between the app and it's companion using a socket based communications channel. This API has an identical implementation in the Device API and the Companion API, so code examples work the same in both locations."

GraphQL Subscriptions

- Allows **subscription to events** using the **Subscription operation** type
- **Transport protocol agnostic** however **popular** implementations based on **Server-Sent events** (e.g. Sangria) and **WebSockets** (e.g. Apollo)
- A subscription is a **read only push stream** (not bidirectional)
- **Subscriptions** are also **client-driven** meaning the consuming app defines what data to include in the event
- **Requires** a **Pub/Sub** system as **events generated** by **mutations** (resolvers) are **captured** and **pushed**
- **GraphQL subscriptions** can be designed/defined with the **GraphQL Schema Definition Language (SDL)***

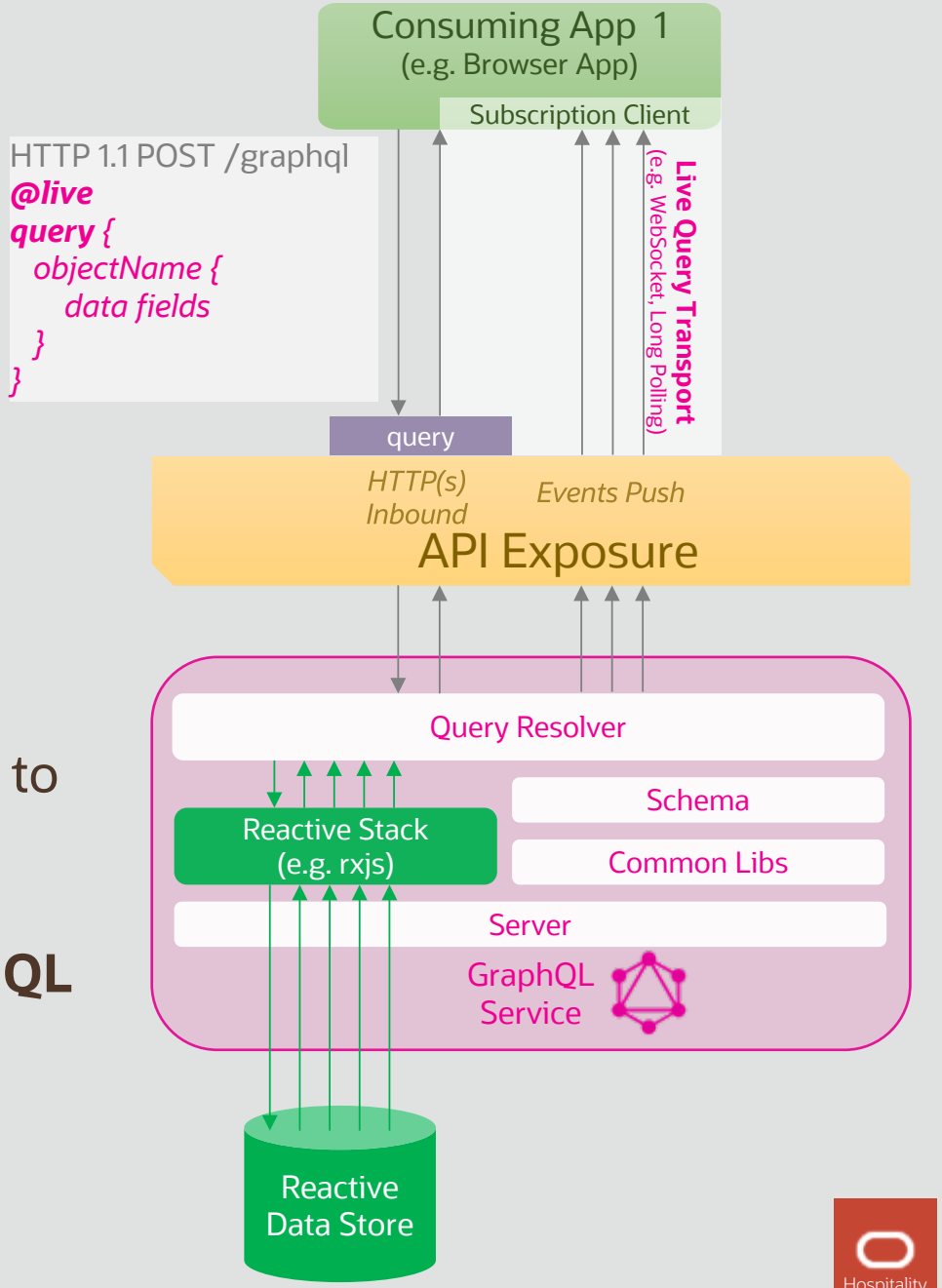
```
HTTP 1.1 POST /graphql
subscription {
  EventName {
    data fields
  }
}
```



* **AsyncAPISpec** working on samples
* **supermodel.io** can be used as as domain (ubiquitous) modelling language

GraphQL Live Queries

- **Transport** protocol **agnostic**
- **Any query** could potentially be a **live query** by using the **@live directive**
- A live query is a **read only push stream** (not bidirectional)
- **Requires** the implementation of a **reactive data layer** (e.g. to tail a query) in the **graphql server**
- Although **queries** can be designed/defined with the **GraphQL Schema Definition Language (SDL),** live queries are not formally defined in the spec (a directive used instead)*



* **supermodel.io** can be used as as domain (ubiquitous) modelling language

GraphQL Subscription & Live Query Samples

- **Hasura.io Graphql Subscriptions & Live Queries:** https://hasura.io/all-features

  "Build powerful applications with GraphQL & Serverless"

- **AWS AppSync:** https://aws.amazon.com/appsync/

  "AppSync simplifies application development by letting you create a flexible API to securely access, manipulate, and combine data from one or more data sources. AppSync is a managed service that uses GraphQL to make it easy for applications to get exactly the data they need"

- **Reactive (Live) Queries at Facebook:** https://www.youtube.com/watch?v=BSw05rJaCpA

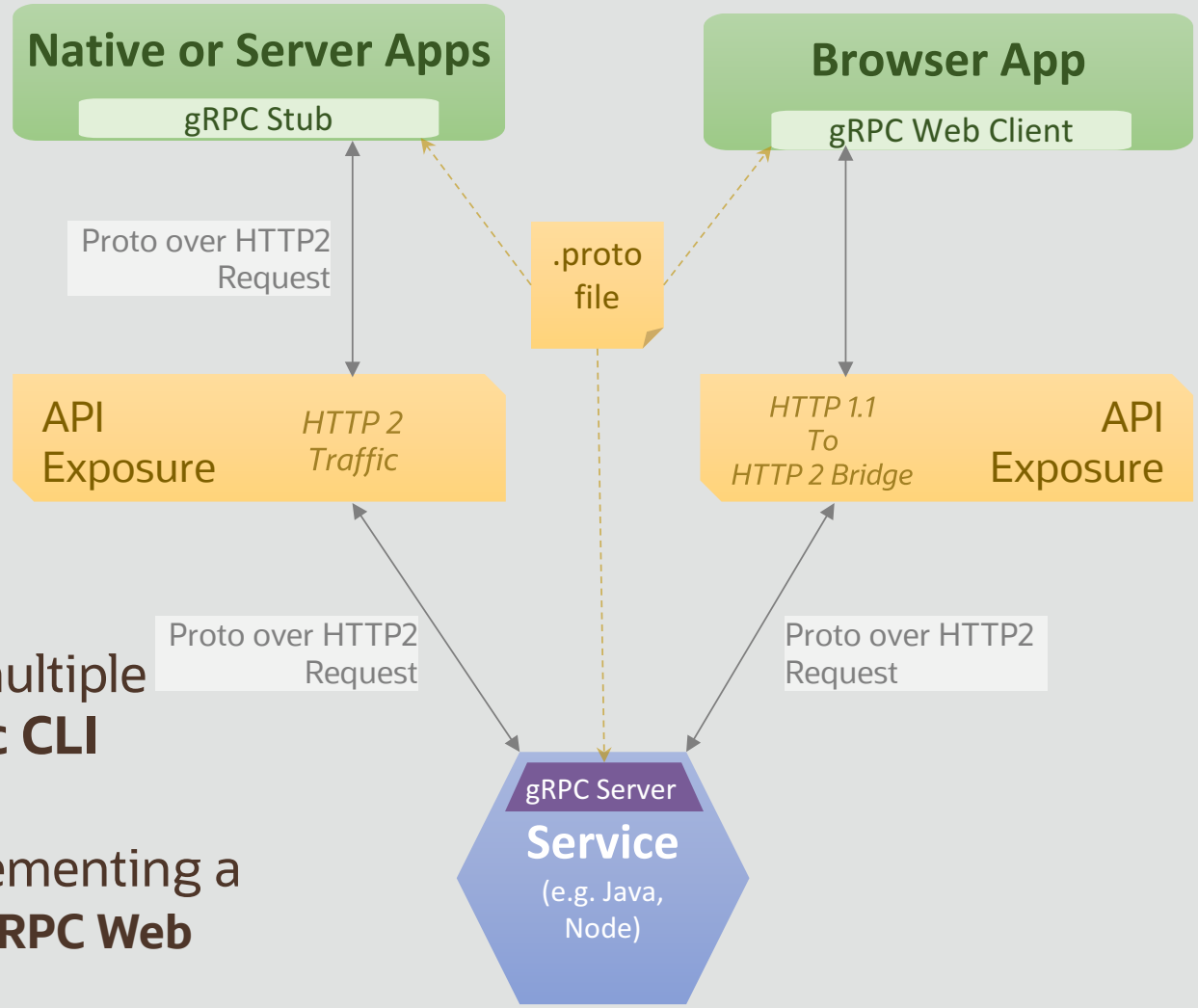
  "At Facebook, we've been developing a new interaction model to enable live GraphQL queries. Leveraging reactive backends and implicit dependency capture, live queries enable a developer-friendly and efficient means for keeping data on clients up to date"

- **Samsara Live Queries:** https://www.youtube.com/watch?v=g-asVW9JFPw

  "At Samsara, we've used live queries in production for the last two years to render live data in our applications by default with minimal boilerplate and plumbing. We'll discuss our experiences: reactive backends, developer happiness, and how you too can add live queries to an existing system"

gRPC

- Makes use of **HTTP/2** as transport protocol
- **Protocol buffers** over HTTP/2 to **serialise / deserialise data** and define the **service interface** (**.proto** file).
- Supports **unary**, **client/server streaming** and full **bidirectional** communication
- gRPC **servers** and **stubs** can be **generated** (in multiple languages) **from the .proto** file using the **protoc CLI**
- Communication over **HTTP 1.1 possible** by implementing a **HTTP 1.1 to HTTP 2 proxy** (e.g. envoy) and the **gRPC Web Client** library

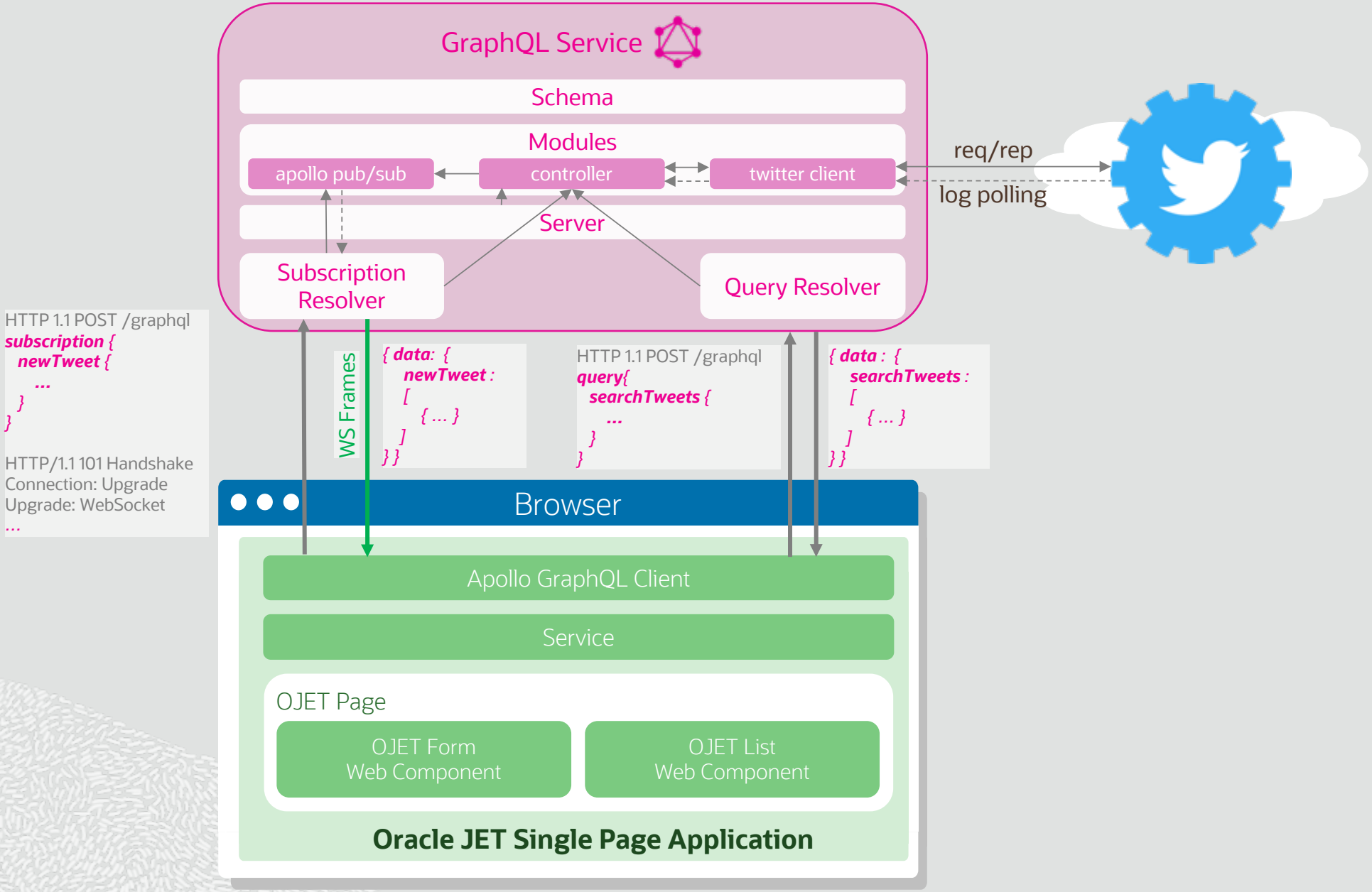


Agenda

1. Sync vs Async APIs
2. Asynchronous on the web?
   Trendy Async API Styles
3. **Demo**
4. Conclusions & Q & A

Hospitality

Demo



https://github.com/luisw19/graphql-subscriptions/tree/master/sample-2

Agenda

1. Sync vs Async APIs
2. Asynchronous on the web?
   Trendy Async API Styles
3. Demo
4. **Conclusions & Q & A**

Hospitality

Comparison (subjective)

(++) Brilliant
(+) Good
(~) Neutral / depends on other factors
(-) Not very good / partly supported
(--) Bad / not supported

	Service to Browser	Service to Service (Internet)	Service to Service (Internal)	Client Driven Contracts	Full Duplex Transport	Learning Curve	Broad Adoption
Long Polling	+	+	~	-	--	+	++
Webhooks	--	++	+	-	--	++	++
WebSockets	++	-	-	-	++	+	+
GraphQL Live Queries	++	-	-	++	--	--	-
GraphQL Subscriptions	++	-	-	++	--	-	-
GRPC	-	-	++	--	++	-	+-

Questions?

Thank you

Luis Weir

Oracle

Hospitality